PowerShell Guru

Meister der Verwaltung von AD, Exchange, SharePoint und SQL

Michael Lappenbusch
FACHINFORMATIKER ANWENDUNGSENTWICKLUNG

Inhaltsverzeichnis

1.Einführung in PowerShell 2
Was ist PowerShell? 2
Unterschiede zwischen PowerShell und Befehlszeile 3
Einführung in die PowerShell-Syntax 4
2.Grundlegende Befehle 5
Navigieren im Dateisystem 5
Anzeigen und Bearbeiten von Dateien 7
Verwalten von Prozessen 8
3.Fortgeschrittene Befehle 9
Verwalten von Benutzer- und Gruppenkonten 9
Verwalten von Paketen und Diensten 10
Verwalten von Netzwerkeinstellungen 11
4.PowerShell-Skripting 13
Einführung in die PowerShell-Skriptsprache 13
Erstellen von Skripten 14
Automatisieren von Aufgaben 15
5.MS Windows-Systemadministration 16
Verwalten von Sicherheitseinstellungen 16
Verwalten von Storage 17
Überwachung und Fehlerbehebung 18
6.Erweiterte Themen 19
Verwalten von Active Directory 19
Verwalten von Exchange 20
Verwalten von SharePoint 21
Verwalten von SQL Server 22
7.Schlußfolgerungen 23
Zusammenfassung der wichtigsten Befehle 23
Tipps und Tricks für erfahrene Anwender 24
Impressum 26

1.Einführung in PowerShell

Was ist PowerShell?

PowerShell ist ein Framework von Microsoft, das es Administratoren ermöglicht, Befehle und Skripte in einer Windows-Umgebung auszuführen. Es basiert auf der .NET Framework und erweitert die Möglichkeiten der Batch-Skriptsprache, die in Windows enthalten ist.

PowerShell bietet eine Kommandozeilen-Schnittstelle (CLI) und eine Skriptsprache, die es ermöglicht, Aufgaben automatisch auszuführen, die sonst manuell ausgeführt werden müssten. Es unterstützt auch die Verwaltung von Remote-Computern und die Verwaltung von Windows-Diensten, Gruppenrichtlinien, Active Directory und vielem mehr.

Ein wichtiger Aspekt von PowerShell ist die Möglichkeit, mit anderen Anwendungen und Diensten zu interagieren, indem Befehle und Skripte verwendet werden, die diese Anwendungen und Dienste unterstützen. Beispielsweise kann PowerShell verwendet werden, um Befehle an Microsoft Exchange auszuführen, um Postfächer und Nachrichten zu verwalten.

PowerShell unterstützt auch die Erstellung von Cmdlets, die sind spezielle Befehle, die mit PowerShell verwendet werden können. Diese Cmdlets können von Drittanbietern oder von Microsoft selbst erstellt werden und erweitern die Funktionalität von PowerShell.

PowerShell wird seit 2006 von Microsoft entwickelt und wird regelmäßig aktualisiert, um neue Funktionen und Verbesserungen zu enthalten. Es ist auf allen aktuellen Versionen von Windows verfügbar, einschließlich Windows 7, Windows 8 und Windows 10.

Unterschiede zwischen PowerShell und Befehlszeile

Die Befehlszeile, auch als "Command Prompt" oder "cmd.exe" bekannt, ist eine Kommandozeilen-Schnittstelle, die in Windows enthalten ist und es Benutzern ermöglicht, Befehle direkt an das Betriebssystem zu senden. Im Gegensatz dazu ist PowerShell ein erweitertes Framework, das auf der Befehlszeile aufbaut und zusätzliche Funktionalität und Skriptfähigkeiten bereitstellt.

Ein wichtiger Unterschied zwischen PowerShell und der Befehlszeile besteht darin, dass PowerShell auf der .NET Framework basiert, während die Befehlszeile auf der Batch-Skriptsprache basiert. Dies ermöglicht PowerShell, mit anderen Anwendungen und Diensten auf tieferer Ebene zu interagieren und komplexere Aufgaben auszuführen.

PowerShell bietet auch eine umfassendere Skriptsprache, die es ermöglicht, Aufgaben automatisch auszuführen und Workflows zu automatisieren. Es unterstützt auch die Verwaltung von Remote-Computern und die Verwaltung von Windows-Diensten, Gruppenrichtlinien, Active Directory und vielem mehr.

Ein weiterer Unterschied besteht darin, dass PowerShell eine umfangreichere Syntax hat, die es ermöglicht, komplexe Aufgaben auszuführen und mehr Kontrolle über das Betriebssystem und die verwalteten Ressourcen zu haben. PowerShell unterstützt auch die Verwendung von Cmdlets, die spezielle Befehle sind, die mit PowerShell verwendet werden können.

Zusammenfassend lässt sich sagen, dass PowerShell eine erweiterte Version der Befehlszeile ist, die zusätzliche Funktionalität und Skriptfähigkeiten bereitstellt und es Administratoren ermöglicht, Aufgaben automatisch auszuführen und Workflows zu automatisieren. Es ist ein mächtiges Framework, das es ermöglicht, tiefer in das Betriebssystem einzudringen und komplexere Aufgaben auszuführen.

Einführung in die PowerShell-Syntax

Die PowerShell-Syntax basiert auf einer Kombination von Textbefehlen und speziellen Operatoren, die es ermöglichen, Aufgaben auszuführen und Daten zu verarbeiten. Einige grundlegende Elemente der PowerShell-Syntax sind:

Befehle: PowerShell-Befehle werden als Cmdlets (kurz für "Command-lets") bezeichnet und sind spezielle Befehle, die mit PowerShell verwendet werden können. Cmdlets haben eine bestimmte Syntax, die aus einem Verb und einem Nomen besteht, z.B. "Get-Process" (um Prozesse auf einem Computer anzuzeigen).

Operatoren: PowerShell unterstützt verschiedene Arten von Operatoren, wie z.B. arithmetische Operatoren (+, -, *, /) oder Vergleichsoperatoren (=, -lt, -gt). Diese Operatoren ermöglichen es, Daten zu verarbeiten und Vergleiche auszuführen.

Variablen: PowerShell unterstützt die Verwendung von Variablen, die es ermöglichen, Daten zwischenzuspeichern und zu verarbeiten. Variablen beginnen mit einem Dollarzeichen ($), gefolgt von einem Namen, z.B. $variable.

Parameter: PowerShell-Cmdlets unterstützen oft Parameter, die es ermöglichen, bestimmte Eigenschaften oder Optionen eines Befehls festzulegen. Parameter werden durch ein Bindestrichzeichen (-) gefolgt vom Namen des Parameters angegeben, z.B. "-Name" oder "-Verbose".

Pipeline: PowerShell ermöglicht es, Befehle zusammenzuführen und die Ausgabe eines Befehls als Eingabe für einen anderen Befehl zu verwenden. Dies wird als "Pipeline" bezeichnet und ermöglicht es, komplexere Aufgaben auszuführen und Daten zu verarbeiten, indem Befehle ineinander verschachtelt werden.

Skripte: PowerShell unterstützt die Erstellung von Skripten, die es ermöglichen, mehrere Befehle automatisch auszuführen. Skripte können in einer Textdatei gespeichert werden und können später ausgeführt werden, indem sie in PowerShell geladen werden.

Es gibt viele weitere Aspekte der PowerShell-Syntax, wie z.B. die Verwendung von Funktionen und Schleifen, die es ermöglichen, komplexere Aufgaben auszufüzuführen und Daten zu verarbeiten. Es gibt auch viele verschiedene Möglichkeiten, wie man die Ausgabe von Befehlen formatieren und filtern kann, um die gewünschten Informationen leichter zugänglich zu machen.

Eine wichtige Sache, die es zu beachten gilt, ist die Unterscheidung zwischen Groß- und Kleinschreibung in PowerShell. Obwohl PowerShell Groß- und Kleinschreibung in der Regel ignoriert, gibt es einige Ausnahmen, insbesondere bei Variablen und Parametern. Es ist daher ratsam, sich an die korrekte Schreibweise zu halten, um Fehler zu vermeiden.

Es gibt auch viele Ressourcen, die verfügbar sind, um PowerShell-Benutzern zu helfen, ihre Kenntnisse zu vertiefen und neue Fähigkeiten zu erlernen. Microsoft stellt zum Beispiel umfangreiche Dokumentation und Leitfäden zur Verfügung, die detailliert erklären, wie man die verschiedenen Aspekte von PowerShell verwendet. Es gibt auch viele Online-Communitys und Foren, die Benutzern dabei helfen können, Probleme zu lösen und Fragen zu beantworten.

Insgesamt ist PowerShell eine mächtige und vielseitige Sprache, die es Administratoren ermöglicht, Aufgaben automatisch auszuführen und Workflows zu automatisieren. Durch eine gründliche Einführung in die PowerShell-Syntax und durch regelmäßiges Üben und Lernen können Benutzer ihre Fähigkeiten verbessern und effektiver mit dem Framework arbeiten.

2.Grundlegende Befehle

Navigieren im Dateisystem

Navigieren im Dateisystem ist eine der grundlegenden Aufgaben, die in PowerShell ausgeführt werden können. PowerShell unterstützt verschiedene Befehle und Operatoren, die es ermöglichen, durch das Dateisystem zu navigieren und Dateien und Ordner zu verwalten. Einige wichtige Befehle, die zum Navigieren im Dateisystem verwendet werden, sind:

Get-ChildItem (alias "dir" oder "ls"): Dieser Befehl zeigt die Unterordner und Dateien in einem angegebenen Pfad an. Der Befehl kann mit verschiedenen Parametern verwendet werden, um bestimmte Dateien oder Ordner zu filtern oder die Ausgabe zu formatieren.

Set-Location (alias "cd"): Dieser Befehl ändert den aktuellen Arbeitsverzeichnis. Der Befehl kann mit einem Pfadangabe verwendet werden, um in ein bestimmtes Verzeichnis zu wechseln.

New-Item: Dieser Befehl erstellt einen neuen Ordner oder eine neue Datei im aktuellen Verzeichnis oder in einem angegebenen Pfad.

Remove-Item: Dieser Befehl löscht einen bestimmten Ordner oder eine bestimmte Datei aus dem Dateisystem. Der Befehl kann mit verschiedenen Parametern verwendet werden, um bestimmte Dateien oder Ordner zu filtern.

Copy-Item: Dieser Befehl kopiert einen bestimmten Ordner oder eine bestimmte Datei in ein anderes Verzeichnis.

Move-Item: Dieser Befehl verschiebt einen bestimmten Ordner oder eine bestimmte Datei in ein anderes Verzeichnis.

Es gibt viele weitere Befehle und Operatoren, die zum Navigieren im Dateisystem verwendet werden können, wie zum Beispiel:

Test-Path: Dieser Befehl überprüft, ob ein bestimmter Pfad im Dateisystem vorhanden ist und gibt einen Boolean-Wert zurück.

Get-Item: Dieser Befehl gibt Informationen über einen bestimmten Pfad im Dateisystem zurück, wie zum Beispiel den vollständigen Pfad, die Größe der Datei oder den Erstellungszeitpunkt.

Get-Acl: Dieser Befehl gibt die Zugriffssteuerungsliste (ACL) eines bestimmten Pfads im Dateisystem zurück.

Set-Acl: Dieser Befehl ändert die Zugriffssteuerungsliste (ACL) eines bestimmten Pfads im Dateisystem.

Compare-Object: Dieser Befehl vergleicht zwei Pfade im Dateisystem und gibt die Unterschiede zwischen ihnen zurück.

Um im Dateisystem zu navigieren, können Sie entweder die vollständige Pfadangabe angeben oder relative Pfade verwenden, die sich auf den aktuellen Arbeitspfad beziehen. Sie können auch Wildcards verwenden, um Dateien oder Ordner zu filtern, und verschiedene Parameter verwenden, um die Ausgabe zu formatieren oder bestimmte Eigenschaften anzuzeigen.

Es ist wichtig zu beachten, dass PowerShell-Befehle und -Skripte, die mit dem Dateisystem arbeiten, ein hohes Risiko für Datenverlust darstellen können, insbesondere wenn sie nicht korrekt verwendet werden. Daher sollten Sie immer eine Sicherung der Daten machen bevor Sie Befehle oder Skripte ausführen und es ist empfehlenswert, die Befehle zunächst in einer Testumgebung auszuführen.

Anzeigen und Bearbeiten von Dateien

In PowerShell können Sie Dateien anzeigen und bearbeiten, indem Sie Befehle und Operatoren verwenden, die speziell für diesen Zweck entwickelt wurden. Einige wichtige Befehle, die zum Anzeigen und Bearbeiten von Dateien verwendet werden, sind:

Get-Content (alias "gc"): Dieser Befehl zeigt den Inhalt einer bestimmten Datei an. Der Befehl kann mit verschiedenen Parametern verwendet werden, um bestimmte Zeilen oder Abschnitte anzuzeigen oder die Ausgabe zu formatieren.

Set-Content (alias "sc"): Dieser Befehl ersetzt den Inhalt einer bestimmten Datei oder fügt neue Inhalte hinzu. Der Befehl kann mit einer Textzeichenfolge oder einem Array von Textzeilen verwendet werden.

Add-Content (alias "ac"): Dieser Befehl fügt eine neue Zeile oder mehrere Zeilen Text zu einer bestimmten Datei hinzu. Der Befehl kann mit einer Textzeichenfolge oder einem Array von Textzeilen verwendet werden.

Out-File: Dieser Befehl schreibt die Ausgabe eines anderen Befehls in eine bestimmte Datei. Der Befehl kann verwendet werden, um die Ausgabe von Cmdlets in eine Datei zu schreiben, anstatt sie auf dem Bildschirm anzuzeigen.

Select-String: Dieser Befehl sucht in einer bestimmten Datei oder einem Ordner nach einer bestimmten Zeichenfolge und gibt die Zeilen zurück, in denen die Zeichenfolge gefunden wurde. Der Befehl kann mit verschiedenen Parametern verwendet werden, um die Suche zu verfeinern.

Get-ItemProperty: Dieser Befehl gibt die Eigenschaften einer bestimmten Datei zurück, wie zum Beispiel den Erstellungszeitpunkt, die Größe der Datei oder die Zugriffsrechte.

Es ist wichtig zu beachten, dass die Befehle und Skripte, die mit dem Bearbeiten von Dateien arbeiten, ein hohes Risiko für Datenverlust darstellen können, insbesondere wenn sie nicht korrekt verwendet werden. Daher sollten Sie immer eine Sicherung der Daten machen bevor Sie Befehle oder Skripte ausführen und es ist empfehlenswert, die Befehle zunächst in einer Testumgebung auszuführen. Es ist auch wichtig, dass Sie die Berechtigungen der Dateien überprüfen, bevor Sie versuchen, diese zu bearbeiten, um sicherzustellen, dass Sie die erforderlichen Berechtigungen haben, um die Dateien zu öffnen und zu bearbeiten.

Es ist auch wichtig, sich bewusst zu sein, dass einige Befehle zum Bearbeiten von Dateien nicht rückgängig gemacht werden können und dass eine unbeabsichtigte Änderung oder Löschung von Dateien zu Datenverlust führen kann. Daher ist es ratsam, vorsichtig zu sein und sicherzustellen, dass Sie genau wissen, was Sie tun, bevor Sie Befehle ausführen, die das Dateisystem verändern.

Es gibt auch viele andere Tools und Anwendungen, die verfügbar sind, um Dateien anzuzeigen und zu bearbeiten, die in PowerShell integriert sind, wie zum Beispiel die Windows-Editor-Anwendungen wie Notepad oder Microsoft Word und es gibt auch Third-Party-Tools die man verwenden kann. Wenn Sie jedoch die volle Kontrolle und Automatisierung benötigen, ist PowerShell eine hervorragende Wahl.

Verwalten von Prozessen

In PowerShell können Sie Prozesse verwalten, indem Sie Befehle und Operatoren verwenden, die speziell für diesen Zweck entwickelt wurden. Einige wichtige Befehle, die zum Verwalten von Prozessen verwendet werden, sind:

Get-Process (alias "ps"): Dieser Befehl zeigt Informationen über die laufenden Prozesse auf einem Computer an. Der Befehl kann mit verschiedenen Parametern verwendet werden, um bestimmte Prozesse zu filtern oder die Ausgabe zu formatieren.

Start-Process (alias "start"): Dieser Befehl startet einen neuen Prozess auf einem Computer. Der Befehl kann mit einem Pfadangabe verwendet werden, um eine bestimmte Anwendung zu starten.

Stop-Process (alias "stop"): Dieser Befehl beendet einen bestimmten Prozess auf einem Computer. Der Befehl kann mit dem Namen oder der ID des Prozesses verwendet werden.

Suspend-Process: Dieser Befehl pausiert einen bestimmten Prozess, was bedeutet, dass der Prozess nicht mehr ausgeführt wird, aber seinen Zustand behält. Der Befehl kann mit dem Namen oder der ID des Prozesses verwendet werden.

Resume-Process: Dieser Befehl setzt einen zuvor pausierten Prozess fort. Der Befehl kann mit dem Namen oder der ID des Prozesses verwendet werden.

Wait-Process: Dieser Befehl blockiert die Ausführung des aktuellen Skripts, bis ein bestimmter Prozess beendet ist. Der Befehl kann mit dem Namen oder der ID des Prozesses verwendet werden.

Get-WmiObject: Dieser Befehl ermöglicht es Ihnen, auf WMI-Objekte (Windows Management Instrumentation) zuzugreifen, die Informationen über Prozesse und andere Systemressourcen enthalten.

Es ist wichtig zu beachten, dass einige der Befehle, die zum Verwalten von Prozessen verwendet werden, möglicherweise Auswirkungen auf den Computer haben können, insbesondere wenn sie nicht korrekt verwendet werden. Bevor Sie einen Prozess beenden, sollten Sie sicherstellen, dass Sie die Auswirkungen kennen und dass es keine Alternative gibt. Es ist auch wichtig, dass Sie die Berechtigungen überprüfen, bevor Sie versuchen, Prozesse zu starten oder zu beenden, um sicherzustellen, dass Sie die erforderlichen Berechtigungen haben.

3.Fortgeschrittene Befehle

Verwalten von Benutzer- und Gruppenkonten

In PowerShell können Sie Benutzer- und Gruppenkonten verwalten, indem Sie Befehle und Operatoren verwenden, die speziell für diesen Zweck entwickelt wurden. Einige wichtige Befehle, die zum Verwalten von Benutzer- und Gruppenkonten verwendet werden, sind:

New-LocalUser: Dieser Befehl erstellt ein neues lokales Benutzerkonto auf einem Computer. Der Befehl kann verwendet werden, um die Eigenschaften des Benutzerkontos, wie z.B. das Kennwort, zu definieren.

Remove-LocalUser: Dieser Befehl löscht ein bestehendes lokales Benutzerkonto auf einem Computer. Der Befehl kann verwendet werden, um das Benutzerkonto und seine Daten dauerhaft zu entfernen.

New-LocalGroup: Dieser Befehl erstellt eine neue lokale Gruppe auf einem Computer. Der Befehl kann verwendet werden, um die Eigenschaften der Gruppe, wie z.B. den Gruppennamen, zu definieren.

Add-LocalGroupMember: Dieser Befehl fügt einen Benutzer oder eine andere Gruppe einer bestehenden lokalen Gruppe hinzu. Der Befehl kann verwendet werden, um Mitgliedschaften von Benutzer- und Gruppenkonten zu verwalten.

Get-LocalUser: Dieser Befehl gibt Informationen über ein bestimmtes lokales Benutzerkonto auf einem Computer zurück, wie z.B. den Benutzernamen, das Kennwort und die Gruppenmitgliedschaft.

Get-LocalGroup: Dieser Befehl gibt Informationen über eine bestimmte lokale Gruppe auf einem Computer zurück, wie z.B. den Gruppennamen und die Mitglieder.

Es ist wichtig zu beachten, dass diese Befehle nur lokale Benutzer- und Gruppenkonten verwalten, und nicht diejenigen in einer Domäne.

Es ist auch wichtig, dass Sie die Berechtigungen überprüfen, bevor Sie versuchen, Benutzer- und Gruppenkonten zu verwalten, um sicherzustellen, dass Sie die erforderlichen Berechtigungen haben. Einige der Befehle, die zum Verwalten von Benutzer- und Gruppenkonten verwendet werden, können möglicherweise Auswirkungen auf den Computer haben, insbesondere wenn sie nicht korrekt verwendet werden. Beispielsweise kann das Löschen eines Benutzerkontos dazu führen, dass Daten und Einstellungen des Benutzers verloren gehen.

Es ist auch wichtig, sicherzustellen, dass Sie die richtigen Befehle verwenden, um die gewünschte Aktion durchzuführen. Beispielsweise gibt es Befehle wie New-ADUser, die verwendet werden können, um Benutzerkonten in einer Active Directory-Domäne zu erstellen, anstatt New-LocalUser, die nur lokale Konten erstellen kann.

Es ist auch wichtig zu beachten, dass die Sicherheit von Benutzer- und Gruppenkonten von entscheidender Bedeutung ist und dass es wichtig ist, sicherzustellen, dass Kennwörter sicher sind und dass unerwünschte Benutzer keinen Zugriff auf das Konto erhalten. Es ist empfehlenswert, regelmäßig zu überprüfen und zu verwalten Benutzer- und Gruppenkonten um sicherzustellen, dass nur autorisierten Personen Zugang haben und dass ungültige Konten entfernt werden.

Verwalten von Paketen und Diensten

In PowerShell können Sie Pakete und Dienste verwalten, indem Sie Befehle und Operatoren verwenden, die speziell für diesen Zweck entwickelt wurden. Einige wichtige Befehle, die zum Verwalten von Paketen und Diensten verwendet werden, sind:

Get-Service: Dieser Befehl zeigt Informationen über die laufenden Dienste auf einem Computer an. Der Befehl kann mit verschiedenen Parametern verwendet werden, um bestimmte Dienste zu filtern oder die Ausgabe zu formatieren.

Start-Service: Dieser Befehl startet einen bestimmten Dienst auf einem Computer. Der Befehl kann mit dem Namen des Dienstes verwendet werden.

Stop-Service: Dieser Befehl beendet einen bestimmten Dienst auf einem Computer. Der Befehl kann mit dem Namen des Dienstes verwendet werden.

Get-WindowsCapability: Dieser Befehl zeigt Informationen über die verfügbaren Windows-Features und -Komponenten an, die installiert oder entfernt werden können.

Add-WindowsCapability: Dieser Befehl fügt ein bestimmtes Windows-Feature oder eine bestimmte Komponente zu einem Computer hinzu. Der Befehl kann mit dem Namen des Features verwendet werden.

Remove-WindowsCapability: Dieser Befehl entfernt ein bestimmtes Windows-Feature oder eine bestimmte Komponente von einem Computer. Der Befehl kann mit dem Namen des Features verwendet werden.

Es ist wichtig zu beachten, dass einige der Befehle, die zum Verwalten von Paketen und Diensten verwendet werden, möglicherweise Auswirkungen auf den Computer haben können, insbesondere wenn sie nicht korrekt verwendet werden. Bevor Sie einen Dienst beenden oder entfernen Sie ein Feature, sollten Sie sicherstellen, dass Sie die Auswirkungen kennen und dass es keine Alternative gibt. Es ist auch wichtig, dass Sie die Berechtigungen überprüfen, bevor Sie versuchen, Pakete oder Dienste zu installieren, zu entfernen oder zu verwalten, um sicherzustellen, dass Sie die erforderlichen Berechtigungen haben. Es ist auch wichtig, die Abhängigkeiten der verschiedenen Pakete und Dienste zu berücksichtigen, um sicherzustellen, dass die richtigen Pakete und Dienste installiert und entfernt werden.

Es ist auch wichtig zu beachten, dass die Verwaltung von Diensten und Paketen eine wichtige Rolle in der Systemverwaltung und -sicherheit spielt. Es ist wichtig, sicherzustellen, dass nur benötigte Dienste und Pakete installiert sind und dass unerwünschte Dienste und Pakete entfernt werden, um die Sicherheit des Systems zu erhöhen und die Leistung des Systems zu optimieren. Es ist empfehlenswert, regelmäßig die installierten Dienste und Pakete zu überprüfen und zu verwalten, um sicherzustellen, dass das System auf dem neuesten Stand ist und dass es keine veralteten oder unsicheren Komponenten gibt.

Verwalten von Netzwerkeinstellungen

In PowerShell können Sie Netzwerkeinstellungen verwalten, indem Sie Befehle und Operatoren verwenden, die speziell für diesen Zweck entwickelt wurden. Einige wichtige Befehle, die zum Verwalten von Netzwerkeinstellungen verwendet werden, sind:

Get-NetIPConfiguration: Dieser Befehl zeigt Informationen über die IP-Konfiguration eines Computers an, wie z.B. die IP-Adresse, die Subnetzmaske und den Standardgateway.

Set-NetIPInterface: Dieser Befehl ermöglicht es Ihnen, die Eigenschaften einer Netzwerkschnittstelle, wie z.B. die IP-Adresse oder die Subnetzmaske, zu ändern.

Get-NetConnectionProfile: Dieser Befehl zeigt Informationen über die Netzwerkverbindungen eines Computers an, wie z.B. den Namen der Verbindung und den Verbindungsstatus.

Set-DnsClientServerAddress: Dieser Befehl ermöglicht es Ihnen, die DNS-Serveradressen für einen Computer zu ändern.

Disable-NetAdapter: Dieser Befehl deaktiviert eine bestimmte Netzwerkschnittstelle auf einem Computer.

Enable-NetAdapter: Dieser Befehl aktiviert eine bestimmte Netzwerkschnittstelle auf einem Computer.

Get-NetAdapter: Dieser Befehl gibt Informationen über die Netzwerkschnittstellen eines Computers zurück, wie z.B. den Namen, die MAC-Adresse und den Status.

New-NetIPAddress: Dieser Befehl erstellt eine neue IP-Adresse für eine bestimmte Netzwerkschnittstelle.

Remove-NetIPAddress: Dieser Befehl entfernt eine bestimmte IP-Adresse von einer Netzwerkschnittstelle.

Es ist wichtig zu beachten, dass einige der Befehle, die zum Verwalten von Netzwerkeinstellungen verwendet werden, möglicherweise Auswirkungen auf den Computer haben können, insbesondere wenn sie nicht korrekt verwendet werden. Bevor Sie eine Änderung an den Netzwerkeinstellungen vornehmen, sollten Sie sicherstellen, dass Sie die Auswirkungen kennen und dass es keine Alternative gibt. Es ist auch wichtig, dass Sie die Berechtigungen überprüfen, bevor Sie versuchen, Netzwerkeinstellungen zu verwalten, um sicherzustellen, dass Sie die erforderlichen Berechtigungen haben.

Es ist auch wichtig zu beachten, dass die Verwaltung von Netzwerkeinstellungen eine wichtige Rolle in der Systemverwaltung und -sicherheit spielt. Es ist wichtig, sicherzustellen, dass die Netzwerkeinstellungen korrekt konfiguriert sind und dass unerwünschte Änderungen vermieden werden, um die Sicherheit des Systems zu erhöhen und die Leistung des Systems zu optimieren. Es ist empfehlenswert, regelmäßig die Netzwerkeinstellungen zu überprüfen und zu verwalten, um sicherzustellen, dass das System ordnungsgemäß funktioniert und dass es keine Konfigurationsfehler gibt.

4.PowerShell-Skripting

Einführung in die PowerShell-Skriptsprache

PowerShell ist eine Skriptsprache, die es ermöglicht, automatisierte Aufgaben durchzuführen und die Verwaltung von Windows-Systemen zu vereinfachen. Ein PowerShell-Skript ist eine Textdatei, die eine Folge von PowerShell-Befehlen enthält, die automatisch ausgeführt werden, wenn die Datei ausgeführt wird.

Ein PowerShell-Skript beginnt normalerweise mit der Angabe des PowerShell-Interpreters, gefolgt von den Befehlen, die ausgeführt werden sollen. Beispielsweise:

```
#!/usr/bin/powershell

Get-Service
```

In diesem einfachen Beispiel wird der Befehl "Get-Service" ausgeführt, wenn das Skript ausgeführt wird, und die Informationen über die laufenden Dienste auf dem Computer angezeigt.

PowerShell-Skripte können auch Variablen, Schleifen, Bedingungen und Funktionen enthalten, um die Skriptfunktionalität zu erweitern. Beispielsweise kann ein Skript eine Schleife verwenden, um mehrere Benutzerkonten zu erstellen, und Variablen verwenden, um die benötigten Informationen für die Konten zu speichern.

PowerShell-Skripte können auf verschiedene Weise ausgeführt werden, z.B. durch Doppelklick auf die Skriptdatei, durch Ausführen des Befehls "powershell.exe" und Angabe des Skriptpfads als Argument oder durch Ausführen des Befehls "Invoke-Expression" und Angabe des Skriptpfads als Argument. Es ist auch möglich, PowerShell-Skripte als geplante Aufgaben oder als Teil von Workflows zu automatisieren.

Es ist wichtig zu beachten, dass PowerShell-Skripte möglicherweise Auswirkungen auf den Computer haben können, insbesondere wenn sie nicht korrekt geschrieben sind oder wenn sie von unauthorisierten Personen erstellt wurden. Es ist daher wichtig, sicherzustellen, dass Sie vertrauenswürdige Quellen für PowerShell-Skripte verwenden und dass Sie die Skripte vor der Ausführung überprüfen, um sicherzustellen, dass sie nicht gefährlich sind. Es ist auch wichtig, die Berechtigungen zu überprüfen, bevor Sie versuchen, PowerShell-Skripte auszuführen, um sicherzustellen, dass Sie die erforderlichen Berechtigungen haben.

Erstellen von Skripten

Das Erstellen von PowerShell-Skripten erfordert grundlegende Kenntnisse in der PowerShell-Syntax und der Verwendung von PowerShell-Befehlen. Um ein PowerShell-Skript zu erstellen, können Sie einen Texteditor wie Notepad verwenden, um eine neue Textdatei zu erstellen und die gewünschten PowerShell-Befehle hinzuzufügen.

Eine einfache Methode, um mit dem Erstellen von PowerShell-Skripten zu beginnen, ist das Aufzeichnen von Schritten, die manuell durchgeführt werden, und das Konvertieren dieser Schritte in PowerShell-Befehle. Es ist auch hilfreich, bestehende PowerShell-Skripte zu untersuchen und zu verstehen, wie sie funktionieren, um Ideen für eigene Skripte zu bekommen.

Es ist wichtig, dass jedes PowerShell-Skript mit einer sogenannten "Shebang" Zeile beginnt, die angibt, dass es sich um ein PowerShell-Skript handelt. Beispielsweise:

```
#!/usr/bin/powershell
```

Es ist auch wichtig, dass jedes Skript kommentiert wird, um die Funktionsweise des Skripts zu erklären und es leichter zu verstehen und zu warten. Ein guter Kommentar sollte die Zweck des Skripts, die verwendeten Befehle und die Ergebnisse beschreiben.

Wenn Sie Skripte erstellen, sollten Sie auch die Sicherheit berücksichtigen. Es ist wichtig, sicherzustellen, dass das Skript nicht versehentlich gefährliche Aktionen ausführt und dass es nicht von unauthorisierten Personen ausgeführt werden kann. Um dies zu gewährleisten, sollten Sie sicherstellen, dass das Skript nur mit den erforderlichen Berechtigungen ausgeführt werden kann und dass es validiert wird, bevor es ausgeführt wird.

Automatisieren von Aufgaben

Ein weiteres wichtiges Konzept beim Erstellen von PowerShell-Skripten ist die Fehlerbehandlung. Es ist wichtig, das Skript so zu schreiben, dass es mögliche Fehler erkennt und angemessen handhabt. Dies kann durch die Verwendung von Try-Catch-Blöcken und die Überprüfung von Rückgabewerten erreicht werden.

Es ist wichtig zu beachten, dass das Erstellen von PowerShell-Skripten Zeit und Übung erfordert, um die Fähigkeiten und Kenntnisse zu entwickeln, um komplexere Aufgaben mit PowerShell-Skripten zu automatisieren. Eine der wichtigsten Vorteile des Einsatzes von PowerShell-Skripten ist die Möglichkeit, Aufgaben automatisch auszuführen, anstatt sie manuell durchzuführen.

Ein Beispiel für die Automatisierung von Aufgaben ist das Erstellen von Benutzerkonten. Anstatt jedes Benutzerkonto manuell zu erstellen, kann ein PowerShell-Skript verwendet werden, um mehrere Benutzerkonten auf einmal zu erstellen, indem es die entsprechenden Befehle automatisch ausführt. Ein weiteres Beispiel ist das automatische Aufräumen von temporären Dateien auf einem Computer. Dies kann durch ein PowerShell-Skript automatisch durchgeführt werden, anstatt dies manuell zu tun.

PowerShell-Skripte können auch als geplante Aufgaben konfiguriert werden, um Aufgaben zu bestimmten Zeiten automatisch auszuführen. Dies ermöglicht es, tägliche oder wöchentliche Aufgaben wie das Sichern von Daten oder das Aktualisieren von Software automatisch durchzuführen, ohne dass eine manuelle Eingabe erforderlich ist.

Es ist wichtig zu beachten, dass die Automatisierung von Aufgaben möglicherweise Auswirkungen auf den Computer hat und dass es wichtig ist, sicherzustellen, dass die automatisierten Aufgaben korrekt konfiguriert sind und dass sie nicht unerwartete Ergebnisse haben. Es ist auch wichtig, die Berechtigungen zu überprüfen, bevor Sie versuchen, automatisierte Aufgaben auszuführen, um sicherzustellen, dass Sie die erforderlichen Berechtigungen haben.

5.MS Windows-Systemadministration

Verwalten von Sicherheitseinstellungen

PowerShell bietet verschiedene Möglichkeiten, um Sicherheitseinstellungen zu verwalten. Sie können Befehle verwenden, um die Sicherheit von Benutzerkonten, Gruppen, Paketen und Diensten zu überwachen und zu verwalten.

Ein Beispiel für die Verwaltung von Sicherheitseinstellungen ist das Erstellen von Benutzerkonten mit starken Passwörtern. Sie können ein PowerShell-Skript erstellen, das automatisch Benutzerkonten erstellt und Passwörter generiert, die den Anforderungen an starke Passwörter entsprechen.

Ein weiteres Beispiel ist die Überwachung von Sicherheitsereignissen. Sie können Befehle verwenden, um Ereignisse aus dem Ereignisprotokoll zu lesen und zu analysieren, um potenzielle Sicherheitsprobleme zu erkennen.

Sie können auch Befehle verwenden, um die Sicherheit von Diensten und Paketen zu überwachen. Beispielsweise können Sie Befehle verwenden, um zu überprüfen, welche Dienste auf einem Computer ausgeführt werden und ob sie sicher konfiguriert sind. Sie können auch Befehle verwenden, um die Versionen von installierten Paketen zu überprüfen und zu überwachen, ob es verfügbare Sicherheitsupdates gibt.

Ein weiteres wichtiges Konzept bei der Verwaltung von Sicherheitseinstellungen ist die Verwaltung von Berechtigungen. Sie können Befehle verwenden, um die Berechtigungen von Benutzerkonten, Gruppen und Ressourcen zu überprüfen und zu verwalten, um sicherzustellen, dass nur autorisierte Benutzer auf bestimmte Ressourcen zugreifen können.

Es ist wichtig zu beachten, dass die Verwaltung von Sicherheitseinstellungen ein kontinuierlicher Prozess ist und dass es wichtig ist, die Sicherheitseinstellungen regelmäßig zu überprüfen und zu aktualisieren, um sicherzustellen, dass das System sicher ist. Es ist auch wichtig, die Berechtigungen zu überprüfen, bevor Sie versuchen, Sicherheitseinstungen zu verwalten, um sicherzustellen, dass Sie die erforderlichen Berechtigungen haben.

Es ist auch wichtig zu beachten, dass es wichtig ist, PowerShell-Skripte, die Sicherheitseinstellungen verwalten, sorgfältig zu überprüfen und zu validieren, bevor Sie sie ausführen, um sicherzustellen, dass sie nicht gefährlich sind und dass sie die erwarteten Ergebnisse liefern. Es ist auch empfehlenswert, die Ausführung von PowerShell-Skripten, die Sicherheitseinstellungen verwalten, zu protokollieren, um Probleme nachverfolgen und Probleme lösen zu können.

Es gibt auch spezialisierte PowerShell-Module und Tools wie das "Security Compliance Manager" (SCM) von Microsoft, welche dafür entwickelt wurden, um Sicherheitsrichtlinien auf Windows-Systemen zu verwalten und zu überwachen. Diese Tools ermöglichen es, Compliance-Richtlinien auf eine einfache und automatisierte Art zu implementieren und zu verwalten.

Verwalten von Storage

PowerShell bietet eine Vielzahl von Befehlen und Cmdlets, die es ermöglichen, Storage-Ressourcen auf einem Windows-System zu verwalten.

Einige der wichtigsten Cmdlets zur Verwaltung von Storage sind:

Get-Disk: Zeigt Informationen zu allen physischen Datenträgern auf dem Computer an, einschließlich freiem und belegtem Speicherplatz.

New-Partition: Erstellt eine neue Partition auf einem Datenträger.

Get-Volume: Zeigt Informationen zu allen Logical Disk (Volume) auf dem Computer an.

Format-Volume: Formatiert ein Volume mit einem bestimmten Dateisystem.

Get-StoragePool: Zeigt Informationen zu allen Storage Pools auf dem Computer an.

Get-VirtualDisk: Zeigt Informationen zu allen Virtual Disks in einem Storage Pool an.

Mit diesen Cmdlets kann man eine Vielzahl von Aufgaben durchführen, wie z.B. das Erstellen und Löschen von Partitionen, das Formatieren von Volumes, das Erstellen und Verwalten von Storage Pools und Virtual Disks. Es ist auch möglich, mit PowerShell Storage-Ressourcen zu automatisieren und Skripte zu erstellen, die regelmäßig ausgeführt werden, um die Storage-Verwaltung zu vereinfachen.

Es gibt auch eine Reihe von anderen Cmdlets, die verwendet werden können, um spezifischere Aufgaben auszuführen, wie z.B. das Verwalten von RAID-Konfigurationen, das Konfigurieren von Storage-Replikationen und das Verwalten von Speicherklassen.

Es ist wichtig zu beachten, dass einige der oben genannten Cmdlets nur auf Windows Server-Systemen verfügbar sind und nicht auf Windows Client-Systemen.

Überwachung und Fehlerbehebung

PowerShell bietet eine Vielzahl von Befehlen und Cmdlets, die es ermöglichen, die Leistung und Verfügbarkeit von Systemen und Anwendungen zu überwachen und Fehler zu beheben.

Einige der wichtigsten Cmdlets zur Überwachung und Fehlerbehebung sind:

Get-EventLog: Zeigt Ereignisse aus den Windows-Ereignisprotokollen an.

Get-Counter: Zeigt Leistungsdaten von Systemressourcen an.

Get-Service: Zeigt den Status von Diensten auf dem Computer an.

Get-Process: Zeigt laufende Prozesse auf dem Computer an.

Get-WmiObject: Ermöglicht den Zugriff auf WMI-Objekte, um Informationen zu Systemressourcen abzurufen.

Test-Connection: Testet die Verbindung zu einem Remote-Computer.

Mit diesen Cmdlets kann man eine Vielzahl von Aufgaben durchführen, wie z.B. das Überwachen von Ereignissen in den Windows-Ereignisprotokollen, das Abfragen von Leistungsdaten von Systemressourcen, das Überwachen von Diensten und Prozessen, das Abfragen von Informationen über WMI-Objekte und das Testen von Netzwerkverbindungen.

Es ist auch möglich, mit PowerShell Überwachungsregeln und -richtlinien zu erstellen, um bestimmte Ereignisse oder Leistungsdaten automatisch zu überwachen und zu melden.

Es gibt auch eine Reihe von anderen Cmdlets, die verwendet werden können, um spezifischere Aufgaben auszuführen, wie z.B. das Abfragen von Informationen über die Sicherheit des Systems oder das Abfragen von Informationen über laufende Transaktionen in Datenbanken.

Es ist wichtig zu beachten, dass einige der oben genannten Cmdlets nur auf Windows Server-Systemen verfügbar sind und nicht auf Windows Client-Systemen.

Es ist auch wichtig darauf zu achten, dass man die richtigen Tools und Cmdlets auswählt um die richtigen Informationen zu erhalten und die Fehlerbehebung zu erleichtern.

6.Erweiterte Themen

Verwalten von Active Directory

Active Directory (AD) ist ein Microsoft-Dienst, der es Administratoren ermöglicht, Benutzer, Computer und andere Ressourcen in einem Netzwerk zu verwalten. AD ermöglicht es, Benutzer und Computer in Domänen zu organisieren, die wiederum in einer Hierarchie organisiert sind.

Um AD mit PowerShell zu verwalten, müssen Sie das Active Directory-Modul für Windows PowerShell installieren. Sobald dies erledigt ist, können Sie PowerShell-Cmdlets verwenden, um verschiedene Aufgaben auszuführen, wie z.B. Benutzerkonten erstellen, Computer hinzufügen oder entfernen, Gruppen erstellen und verwalten und vieles mehr.

Einige Beispiele für häufig verwendete Cmdlets zur Verwaltung von AD mit PowerShell:

New-ADUser: Erstellt ein neues Benutzerkonto in AD

Set-ADUser: Ändert die Eigenschaften eines vorhandenen Benutzerkontos

Remove-ADUser: Löscht ein vorhandenes Benutzerkonto

Get-ADUser: Ruft Informationen zu einem vorhandenen Benutzerkonto ab

New-ADGroup: Erstellt eine neue Gruppe in AD

Add-ADGroupMember: Fügt Mitglieder zu einer vorhandenen Gruppe hinzu

Get-ADGroup: Ruft Informationen zu einer vorhandenen Gruppe ab

Es gibt viele weitere Cmdlets, die Sie verwenden können, um AD mit PowerShell zu verwalten. Es empfiehlt sich, die Hilfe für jedes Cmdlet zu lesen, um sicherzustellen, dass Sie es korrekt verwenden und um zu verstehen, welche Optionen verfügbar sind.

Verwalten von Exchange

Exchange ist ein E-Mail- und Kalendersystem von Microsoft, das es Unternehmen ermöglicht, E-Mail, Kalender, Kontakte und Aufgaben zu verwalten. Exchange bietet auch Funktionen wie die Verwaltung von Regeln und Richtlinien, die Überwachung von E-Mail-Verkehr und die Unterstützung von mobilen Geräten.

Um Exchange mit PowerShell zu verwalten, müssen Sie das Exchange-Modul für Windows PowerShell installieren. Sobald dies erledigt ist, können Sie PowerShell-Cmdlets verwenden, um verschiedene Aufgaben auszuführen, wie z.B. Benutzerkonten erstellen, E-Mail-Adressen verwalten, Postfächer verwalten, Regeln erstellen und vieles mehr.

Einige Beispiele für häufig verwendete Cmdlets zur Verwaltung von Exchange mit PowerShell:

New-Mailbox: Erstellt ein neues Postfach

Set-Mailbox: Ändert die Eigenschaften eines vorhandenen Postfachs

Remove-Mailbox: Löscht ein vorhandenes Postfach

Get-Mailbox: Ruft Informationen zu einem vorhandenen Postfach ab

New-MailboxPermission: Erteilt Berechtigungen für ein Postfach

New-TransportRule: Erstellt eine neue Transportregel

Get-TransportRule: Ruft Informationen zu einer vorhandenen Transportregel ab

Es gibt viele weitere Cmdlets, die Sie verwenden können, um Exchange mit PowerShell zu verwalten. Es empfiehlt sich, die Hilfe für jedes Cmdlet zu lesen, um sicherzustellen, dass Sie es korrekt verwenden und um zu verstehen, welche Optionen verfügbar sind. Es gibt auch einige spezielle Cmdlets, die nur in Exchange Online oder Exchange Server verfügbar sind, so dass es ratsam ist, sich auf die spezifische Version zu beziehen.

Verwalten von SharePoint

SharePoint ist eine Plattform von Microsoft, die es Unternehmen ermöglicht, Dokumente, Webinhalte, Anwendungen und Prozesse zu verwalten und zu teilen. SharePoint bietet auch Funktionen wie die Zusammenarbeit in Echtzeit, die Verwaltung von Zugriffsrechten, die Suche nach Inhalten und die Entwicklung von benutzerdefinierten Lösungen.

Um SharePoint mit PowerShell zu verwalten, müssen Sie das SharePoint-Modul für Windows PowerShell installieren. Sobald dies erledigt ist, können Sie PowerShell-Cmdlets verwenden, um verschiedene Aufgaben auszuführen, wie z.B. die Erstellung von Websites, die Verwaltung von Listen und Bibliotheken, die Verwaltung von Benutzerzugriffsrechten, die Verwaltung von Inhaltsstrukturen und vieles mehr.

Einige Beispiele für häufig verwendete Cmdlets zur Verwaltung von SharePoint mit PowerShell:

New-SPWeb: Erstellt eine neue Website

Set-SPWeb: Ändert die Eigenschaften einer vorhandenen Website

Remove-SPWeb: Löscht eine vorhandene Website

Get-SPWeb: Ruft Informationen zu einer vorhandenen Website ab

New-SPList: Erstellt eine neue Liste

Add-SPListItem: Fügt ein neues Element zu einer vorhandenen Liste hinzu

Get-SPList: Ruft Informationen zu einer vorhandenen Liste ab

Grant-SPObjectSecurity: Erteilt Zugriffsrechte für ein SharePoint-Objekt

Get-SPUser: Ruft Informationen zu einem vorhandenen Benutzer in SharePoint ab

Es gibt viele weitere Cmdlets, die Sie verwenden können, um SharePoint mit PowerShell zu verwalten. Es empfiehlt sich, die Hilfe für jedes Cmdlet zu lesen, um sicherzustellen, dass Sie es korrekt verwenden und um zu verstehen, welche Optionen verfügbar sind. Es ist auch ratsam, sich mit den grundlegenden Konzepten von SharePoint vertraut zu machen, bevor Sie damit beginnen, es mit PowerShell zu verwalten, um das Verständnis der Befehle zu erleichtern.

Verwalten von SQL Server

SQL Server ist eine relationale Datenbankmanagement-System (RDBMS) von Microsoft, die es Unternehmen ermöglicht, Daten zu speichern, abzufragen und zu verwalten. SQL Server bietet auch Funktionen wie die Verwaltung von Sicherheit, die Unterstützung von Business Intelligence (BI) und die Entwicklung von benutzerdefinierten Lösungen.

Um SQL Server mit PowerShell zu verwalten, müssen Sie das SQL Server-Modul für Windows PowerShell installieren. Sobald dies erledigt ist, können Sie PowerShell-Cmdlets verwenden, um verschiedene Aufgaben auszuführen, wie z.B. die Erstellung von Datenbanken, die Verwaltung von Tabellen, die Verwaltung von Benutzerzugriffsrechten, die Verwaltung von Sicherheit und vieles mehr.

Einige Beispiele für häufig verwendete Cmdlets zur Verwaltung von SQL Server mit PowerShell:

Invoke-Sqlcmd: Führt eine Transact-SQL-Abfrage oder ein Skript auf einer SQL Server-Instanz aus.

Backup-SqlDatabase: Erstellt ein Backup einer SQL Server-Datenbank

Restore-SqlDatabase: Stellt eine SQL Server-Datenbank aus einem Backup wieder her

New-SqlLogin: Erstellt ein neues Anmeldeobjekt für SQL Server

Remove-SqlLogin: Löscht ein vorhandenes Anmeldeobjekt

Add-SqlAvailabilityDatabase: Fügt eine Datenbank zu einer Verfügbarkeitsgruppe hinzu

Get-SqlInstance: Ruft Informationen zu einer SQL Server-Instanz ab

Es gibt viele weitere Cmdlets, die Sie verwenden können, um SQL Server mit PowerShell zu verwalten. Es empfiehlt sich, die Hilfe für jedes Cmdlet zu lesen, um sicherzustellen, dass Sie es korrekt verwenden und um zu verstehen, welche Optionen verfügbar sind. Es ist auch ratsam, sich mit den grundlegenden Konzepten von SQL Server vertraut zu machen, bevor Sie damit beginnen, es mit PowerShell zu verwalten, um das Verständnis der Befehle zu erleichtern.

7.Schlußfolgerungen

Zusammenfassung der wichtigsten Befehle

Active Directory:

New-ADUser: Erstellt ein neues Benutzerkonto in AD

Set-ADUser: Ändert die Eigenschaften eines vorhandenen Benutzerkontos

Remove-ADUser: Löscht ein vorhandenes Benutzerkonto

Get-ADUser: Ruft Informationen zu einem vorhandenen Benutzerkonto ab

New-ADGroup: Erstellt eine neue Gruppe in AD

Add-ADGroupMember: Fügt Mitglieder zu einer vorhandenen Gruppe hinzu

Get-ADGroup: Ruft Informationen zu einer vorhandenen Gruppe ab

Exchange:

New-Mailbox: Erstellt ein neues Postfach

Set-Mailbox: Ändert die Eigenschaften eines vorhandenen Postfachs

Remove-Mailbox: Löscht ein vorhandenes Postfach

Get-Mailbox: Ruft Informationen zu einem vorhandenen Postfach ab

New-MailboxPermission: Erteilt Berechtigungen für ein Postfach

New-TransportRule: Erstellt eine neue Transportregel

Get-TransportRule: Ruft Informationen zu einer vorhandenen Transportregel ab

SharePoint:

New-SPWeb: Erstellt eine neue Website

Set-SPWeb: Ändert die Eigenschaften einer vorhandenen Website

Remove-SPWeb: Löscht eine vorhandene Website

Get-SPWeb: Ruft Informationen zu einer vorhandenen Website ab

New-SPList: Erstellt eine neue Liste

Add-SPListItem: Fügt ein neues Element zu einer vorhandenen Liste hinzu

Get-SPList: Ruft Informationen zu einer vorhandenen Liste ab

Grant-SPObjectSecurity: Erteilt Zugriffsrechte für ein SharePoint-Objekt

Get-SPUser: Ruft Informationen zu einem vorhandenen Benutzer in SharePoint ab

SQL Server:

Invoke-Sqlcmd: Führt eine Transact-SQL-Abfrage oder ein Skript auf einer SQL Server-Instanz aus

Backup-SqlDatabase: Erstellt ein Backup einer SQL Server-Datenbank

Restore-SqlDatabase: Stellt eine SQL Server-Datenbank aus einem Backup wieder her

New-SqlLogin: Erstellt ein neues Anmeldeobjekt für SQL Server

Remove-SqlLogin: Löscht ein vorhandenes Anmeldeobjekt

Add-SqlAvailabilityDatabase: Fügt eine Datenbank zu einer Verfügbarkeitsgruppe hinzu

Get-SqlInstance: Ruft Informationen zu einer SQL Server-Instanz ab

Es gibt viele weitere Cmdlets, die verfere Befehle und Optionen, die je nach Bedarf verwendet werden können, um die Verwaltung von Active Directory, Exchange, SharePoint und SQL Server zu automatisieren und zu optimieren. Es ist jedoch wichtig zu beachten, dass die Verwaltung dieser Systeme mit PowerShell erfordert, dass Sie über fundierte Kenntnisse und Erfahrungen in Bezug auf die jeweiligen Systeme und die PowerShell-Sprache verfügen. Eine falsche Verwendung der Befehle kann zu unerwarteten Ergebnissen oder sogar zu Datenverlust führen. Es wird daher empfohlen, sich vor der Verwendung der Befehle mit der Dokumentation und dem technischen Support vertraut zu machen.

Tipps und Tricks für erfahrene Anwender

können, die Effizienz und die Genauigkeit der Verwaltung von Active Directory, Exchange, SharePoint und SQL Server zu verbessern:

Verwenden Sie Aliase: PowerShell hat viele Cmdlets mit langen und komplexen Namen. Sie können Aliase verwenden, um diese Befehle einfacher und schneller aufzurufen.

Verwenden Sie Tab-Vervollständigung: Durch Drücken der Tabulatortaste können Sie schnell die verfügbaren Optionen eines Befehls aufrufen, anstatt sie manuell einzugeben.

Verwenden Sie Cmdlet-Verbundbefehle: PowerShell bietet Cmdlet-Verbundbefehle, die es ermöglichen, mehrere Befehle in einer einzigen Zeile auszuführen. Beispielsweise können Sie den Befehl "Get-ADUser -Filter * | Export-CSV -Path C:\Users.csv" verwenden, um alle Benutzer aus AD in eine CSV-Datei zu exportieren.

Erstellen Sie Skripte: Anstatt jeden Befehl manuell einzugeben, können Sie Skripte erstellen, um wiederkehrende Aufgaben automatisch auszuführen.

Verwenden Sie die Pipeline: die Pipeline ermöglicht es, die Ausgabe eines Befehls als Eingabe für den nächsten Befehl zu verwenden, was die Automatisierung von Aufgaben erleichtert und die Lesbarkeit des Codes verbessert.

Verwenden Sie die Parameter-Tabellensyntax: Sie können mehrere Werte für einen bestimmten Parameter auf einmal angeben, indem Sie eine Tabelle mit Werten verwenden. Beispielsweise können Sie den Befehl "Add-ADGroupMember -Identity "Marketing" -Members @("User1","User2","User3")" verwenden, um mehrere Benutzer auf einmal einer Gruppe hinzuzufügen.

Verwenden Sie die -WhatIf-Option: Diese Option zeigt an, welche Änderungen vorgenommen werden, bevor der Befehl tatsächlich ausgeführt wird, was dazu beiträgt, unbeabsichtigte Auswirkungen zu vermeiden.

Verwenden Sie -Confirm-Option: Mit dieser Option wird der Anwender gefragt, ob er die Ausführung des Befehls bestätigen möchte, bevor dieser ausgeführt wird.

Verwenden Sie die -ErrorAction-Option: Diese Option ermöglicht es, die Art und Weise zu steuern, wie PowerShell auf Fehler reagieren soll.

Verwenden Sie die -Verbose-Option: Diese Option gibt detaillierte Informationen zur Ausführung des Befehls aus, was dazu beiträgt, Probleme schneller zu identifizieren und zu lösen.

Es ist wichtig zu beachten, dass die Verwendung dieser Tipps und Tricks in Kombination mit einem tiefen Verständnis der PowerShell-Sprache und der zugrunde liegenden Systeme, für die Sie verwalten, erforderlich ist. Es ist auch wichtig, sich immer bewusst über die Auswirkungen von Änderungen zu sein und regelmäßig Backups zu erstellen, um Datenverlust zu vermeiden. Es empfiehlt sich auch, die Dokumentation und den technischen Support zu nutzen, um sicherzustellen, dass Sie die bestmögliche Verwaltung erreichen.

Impressum

Dieses Buch wurde unter der
Creative Commons Attribution-NonCommercial-NoDerivatives (CC BY-NC-ND) Lizenz veröffentlicht.

Diese Lizenz ermöglicht es anderen, das Buch kostenlos zu nutzen und zu teilen, solange sie den Autor und die Quelle des Buches nennen und es nicht für kommerzielle Zwecke verwenden.

Autor: **Michael Lappenbusch**

Email: admin@perplex.click

Homepage: https://www.perplex.click

Erscheinungsjahr: 2023

Some of the content comes from: ChatGPT